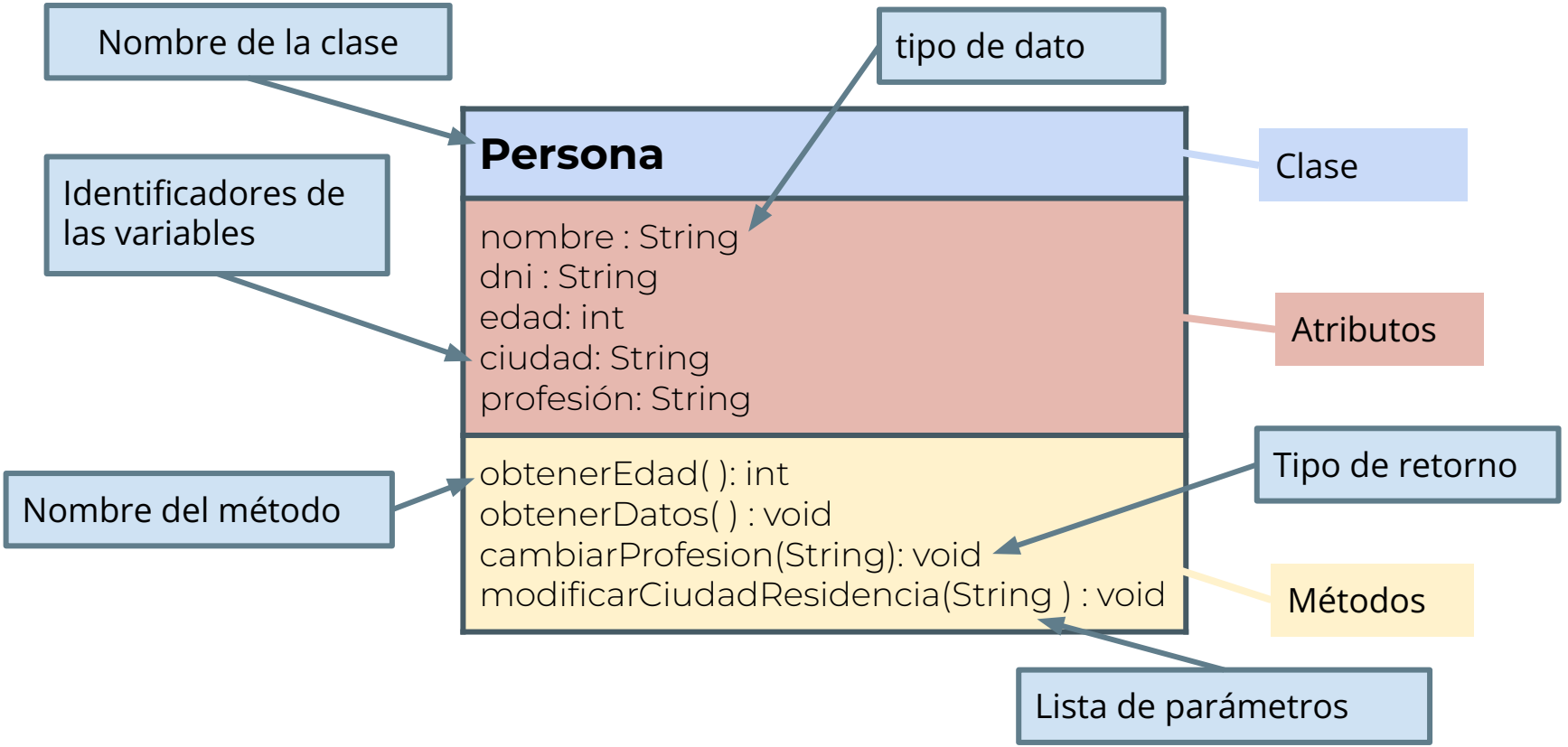
Diseño orientado a objetos - UML

Introducción a UML

- El **UML** (Unified Modelling Language o Lenguaje de Modelado Unificado) se utiliza para establecer cómo se estructura la resolución de un problema mediante la orientación a objetos. Además, se utiliza para saber de qué manera interactúan los diferentes componentes para lograr una tarea concreta.
- El UML es un lenguaje estándar que permite especificar con notación gráfica software orientado a objetos.
- Estas bases son las siguientes:
  - → Todo es un objeto, con una identidad propia.
  - → Un programa es un conjunto de objetos que interactúan entre ellos.
  - → Un objeto puede estar formado por otros objetos más simples.
  - → Cada objeto pertenece a un tipo concreto: una clase.
  - → Objetos del mismo tipo tienen un comportamiento idéntico.

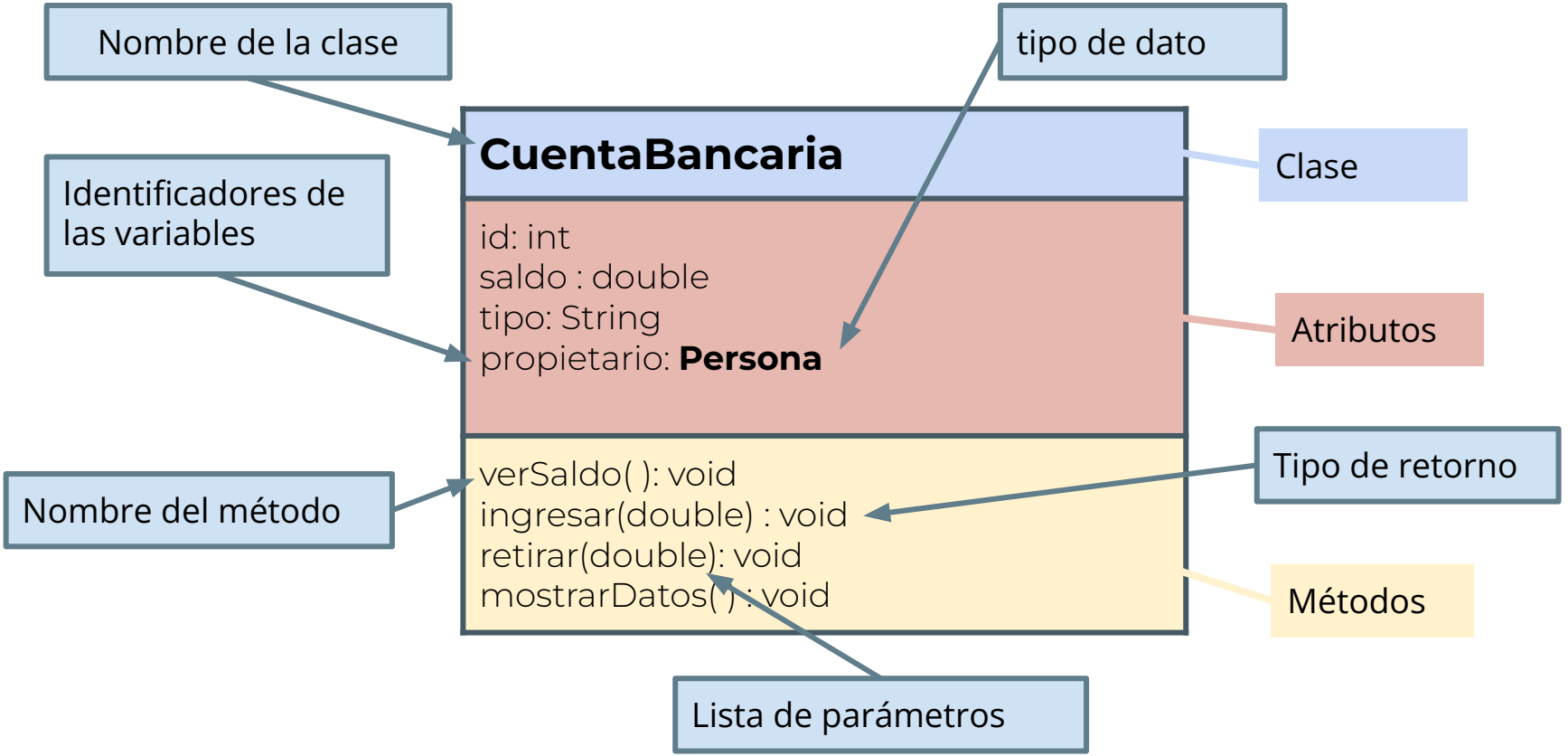
Ejemplo de clase en UML

A continuación, se muestra un ejemplo de la clase **Persona** en UML:



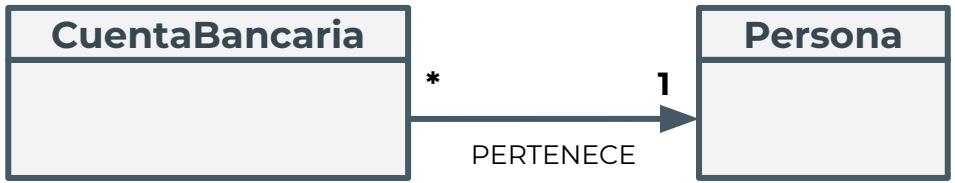
Ejemplo de clase en UML

A continuación, se muestra un ejemplo de la clase **CuentaBancaria** en UML:



Relaciones entre clases

- Una vez identificadas las clases de los objetos que componen el problema a resolver, el siguiente paso es establecer qué relaciones hay entre dichas clases. Cada relación indica que hay una conexión entre los objetos de una clase y los de otra.
- **El tipo de relación más frecuente es la asociación**. Se considera que existe una asociación entre dos clases cuando se quiere indicar que los objetos de una clase pueden llamar operaciones sobre los objetos de otra.



Relaciones entre clases

- Dada una asociación, se debe especificar:
  - → **En el centro, el nombre de la asociación**.
  - → **Con una flecha se especifica la navegabilidad**. Partiendo del nombre de la asociación y los métodos de las clases, se debe poder establecer cuál es la clase **origen** y cuál el **destino**.
  - → **La navegabilidad indica el sentido de las interacciones entre objetos**: a qué clase pertenecen los objetos que pueden llamar operaciones y a qué clase los objetos que reciben estas llamadas.
  - → Si no se especifica navegabilidad(sin flecha), se tratará de una **asociación bidireccional**, ambas clases podrán llamar a métodos de la otra.

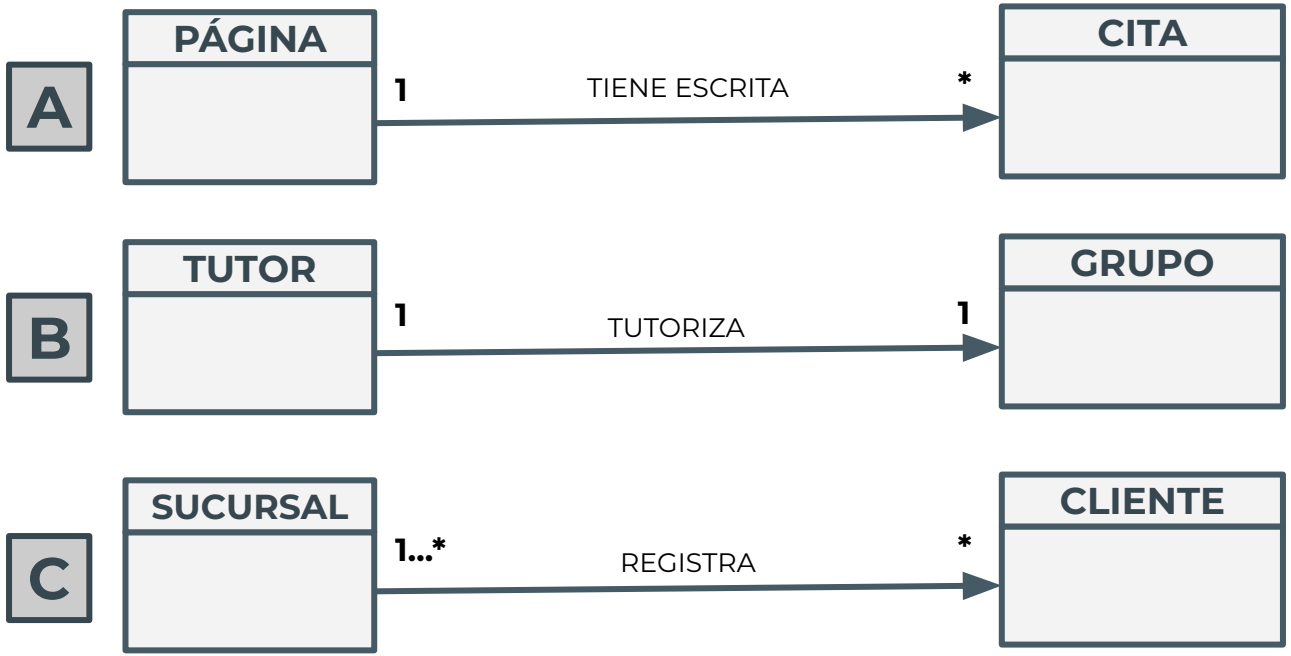
Relaciones entre clases

- **En cada extremo se especifica la cardinalidad**.
- **La cardinalidad debe especificar con cuantas instancias de una de las clases puede estar enlazada una instancia de la otra clase** en un momento determinado de la ejecución del programa.
- Para establecer rangos de valores posibles, **se usan los límites inferior y superior separados con tres puntos**.

Diferentes cardinalidades y su significado	
1	Sólo un enlace
0...1	Ninguno o un enlace
2,4	Dos o cuatro enlaces
1...5	Entre 1 y 5 enlaces
1...*	Entre 1 y un número indeterminado, es decir, más de 1
*	Un número indeterminado. Es equivalente a 0...*

Relaciones entre clases

- La **navegabilidad** y la **cardinalidad** son imprescindibles ya que la decisión que se tome en estos aspectos dentro de la etapa de diseño tendrá implicaciones directas sobre la implementación.



Relaciones entre clases

- Una instancia cualquiera de la clase Página puede enlazar hasta un número indeterminado de objetos diferentes de la clase Cita.
- Dado un objeto cualquiera de la clase Cita, sólo estará enlazado a un único objeto Página. Por lo tanto, no se puede tener una misma cita en dos páginas diferentes (pero sí tener dos citas diferentes y de contenido idéntico, con los mismos valores para los atributos, cada una a una página diferente).
- Tampoco puede haber citas que, a pesar de ser en la aplicación, no estén escritas en ninguna página.

Relaciones entre clases

- Un objeto de la clase **Tutor** siempre tiene un objeto de la clase **Grupo** enlazado. Por tanto, un tutor siempre tutoriza a un grupo.
- No se puede dar el caso de que un tutor no tutorice a ningún grupo. La inversa también es cierta, todo grupo es tutorizado por algún tutor.
- El tutor puede llamar operaciones sobre el grupo, pero no al revés. Esto tiene sentido, ya que es el tutor el que controla al grupo.

Relaciones entre clases

- Dado un cliente, éste puede estar registrado en más de una sucursal, pero al menos siempre lo estará en una. Nunca se puede dar el caso de un cliente dado de alta en el sistema pero que no esté registrado en ninguna sucursal.

Agregación

- Asociación especial mediante la cual los objetos de cierta clase forman parte de los objetos de otra. En el diagrama estático UML, esto se representa gráficamente añadiendo un rombo blanco en el extremo de la asociación donde está la clase que representa el todo.
- Como con este símbolo ya se dice cuál es la relación entre los objetos de ambas clases, se pueden omitir el nombre y la función en los descriptores de la asociación.

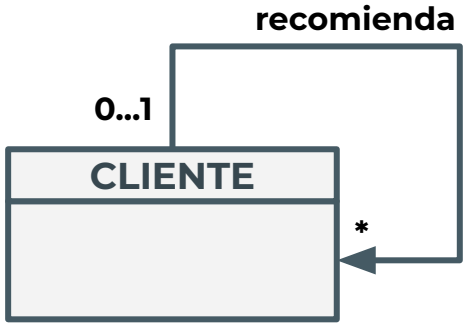
- Una página contiene citas escritas en su interior y se puede considerar que lo escrito en una página es parte de la misma.
- Expresa **contiene o puede contener.** En el ejemplo anterior una página puede contener citas, pero puede existir una página que en principio no tenga citas.

Composición

- Expresa el concepto de **es parte de**, ya que la clase compuesta no tiene sentido sin sus componentes.
- En contraposición, en una agregación, el agregado sí tiene sentido sin ninguno de sus componentes.
- Una composición es una forma de agregación que requiere que los objetos componentes sólo pertenezcan a un único objeto agregado y que, además, este último no exista si no existen los componentes.
- Cualquier asociación en que el diseñador pondría un verbo de el estilo es parte de, será una composición.
- Este tipo de asociación se representa de forma idéntica a una agregación, sólo que en este caso el rombo es de color negro.
    - → No tiene sentido una agenda sin páginas.
    - → Tampoco puede ser que una misma página esté en más de una agenda.
    - → En cambio, sí tiene sentido una página en blanco sin ningún cita, por lo que el caso Página-Cita es una agregación pero no una composición.

Asociación reflexiva

- Una asociación reflexiva es aquella en que la clase origen y el destino son la misma.
- No se puede aplicar una asociación reflexiva a una composición.
- Ocurre cuando hay una asociación entre instancias de una clase.
- Un ejemplo de este caso se muestra en la figura, en el que los clientes de la aplicación de gestión recomiendan otros clientes. Se trata de una asociación reflexiva, ya que tanto quien recomienda como quien es recomendado, un cliente, pertenecen a la misma clase.



Implementar asociaciones entre clases

Implementación de asociaciones en JAVA

- Las asociaciones entre clases se captan mediante atributos en la clase origen de la relación, el tipo de los cuales es la clase destino.
    - Habrá tantos atributos de cada tipo como la cota superior de la cardinalidad en el extremo destino de la relación.
        - Si la cardinalidad es 2 habrán 2 atributos de la clase destino en clase origen.
        - Si cardinalidad 1, habrá un único atributo de la clase destino en la clase origen.
        - Las cardinalidades indeterminadas (caso *), se suele usar cualquier colección que se pueda modificar de forma dinámica, por ejemplo un ArrayList.
- Decidir qué métodos incluir en cada clase puede ser difícil durante el proceso de diseño
- Un principio clave es lograr cohesión, es decir, tener clases coherentes y con un propósito claro en el problema que se está resolviendo. Es importante que las clases trabajen juntas en armonía y que cada una tenga un objetivo específico.
- En una clase, solo se deben incluir los atributos y operaciones esenciales para cumplir su determinada función.

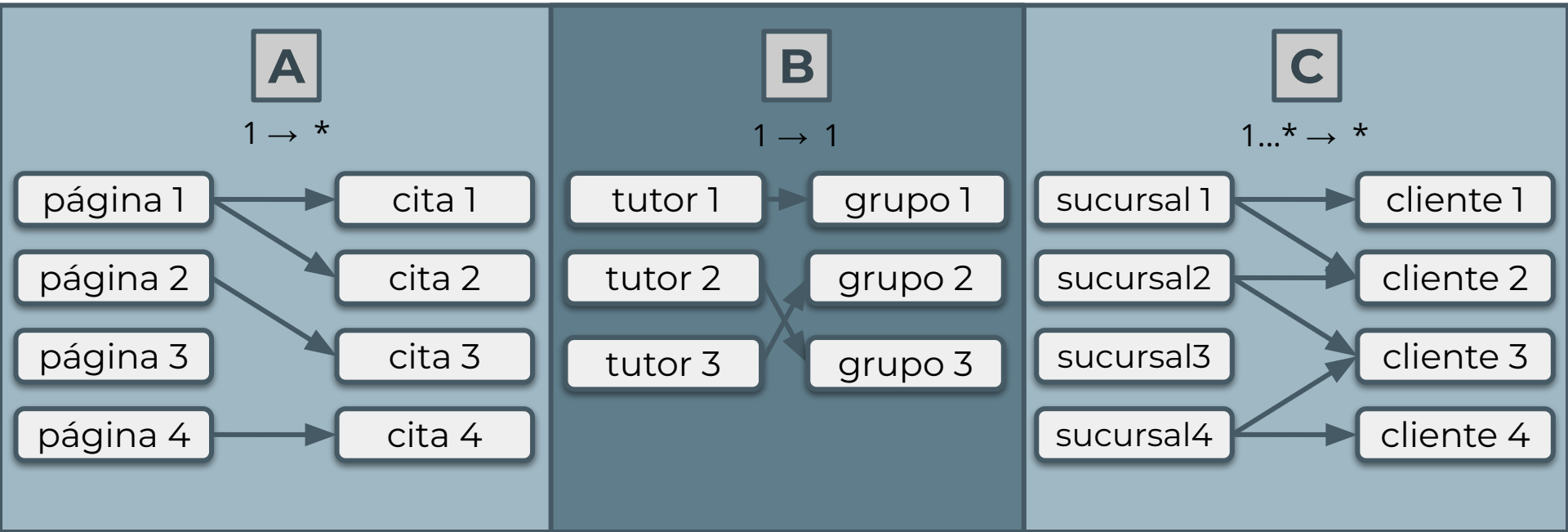
Mapa de objetos

Mapa de objetos

- Se utilizan para reflexionar sobre si una cardinalidad representa lo que el diseñador quiere.
- Se trata de esquemas que representan los diferentes estados posibles de la aplicación.
- Los mapas de objetos sólo son una herramienta de apoyo, y no se utilizan como mecanismo formal para representar el diseño.
- En un **mapa de objetos** se muestran todos los objetos instanciados y los enlaces que hay entre ellos en un momento determinado de la ejecución, la aplicación de acuerdo con lo que se ha representado en el diagrama UML.
    - → Dado un cliente, éste puede estar registrado en más de una sucursal, pero al menos siempre lo estará en una. Nunca se puede dar el caso de un cliente dado de alta en el sistema pero que no esté registrado en ninguna sucursal.

Mapa de objetos

- La figura representa una serie de mapas de objetos, uno por cada caso, con diferentes objetos enlazados correctamente según la cardinalidad especificada en la asociación.
- Los enlaces se representan con flechas según la navegabilidad de las asociaciones.



Mapa de objetos

- La figura muestra algunos casos de estados que se consideran incorrectos según las cardinalidades especificadas en las asociaciones.